Computing Systems Roadmap and its Impact on Software Development

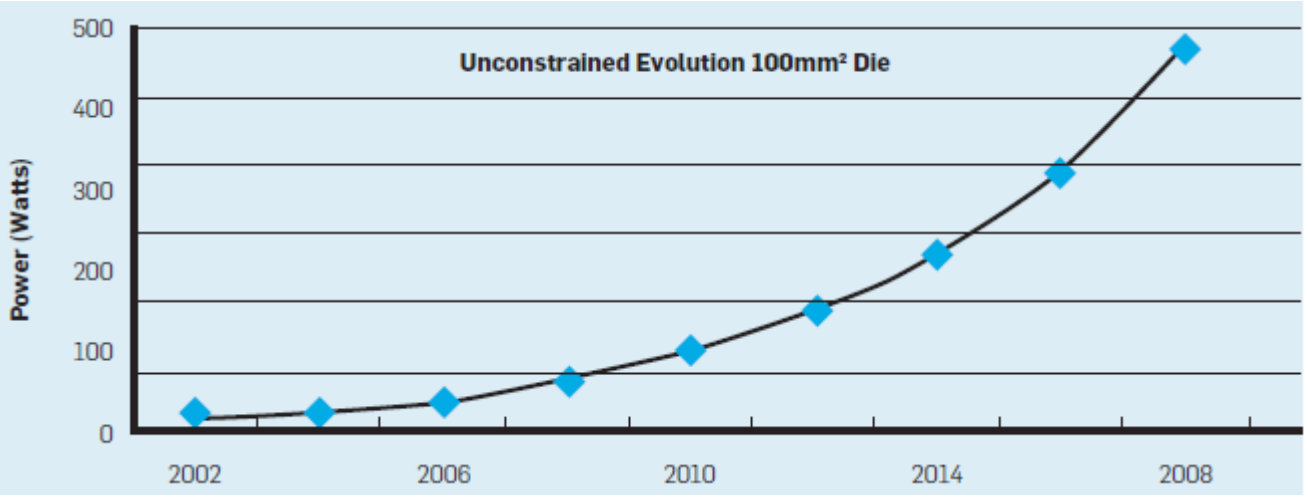
Michael Ernst, BNL

HSF Workshop at SLAC

20-21 January, 2015

Processor Die Scaling

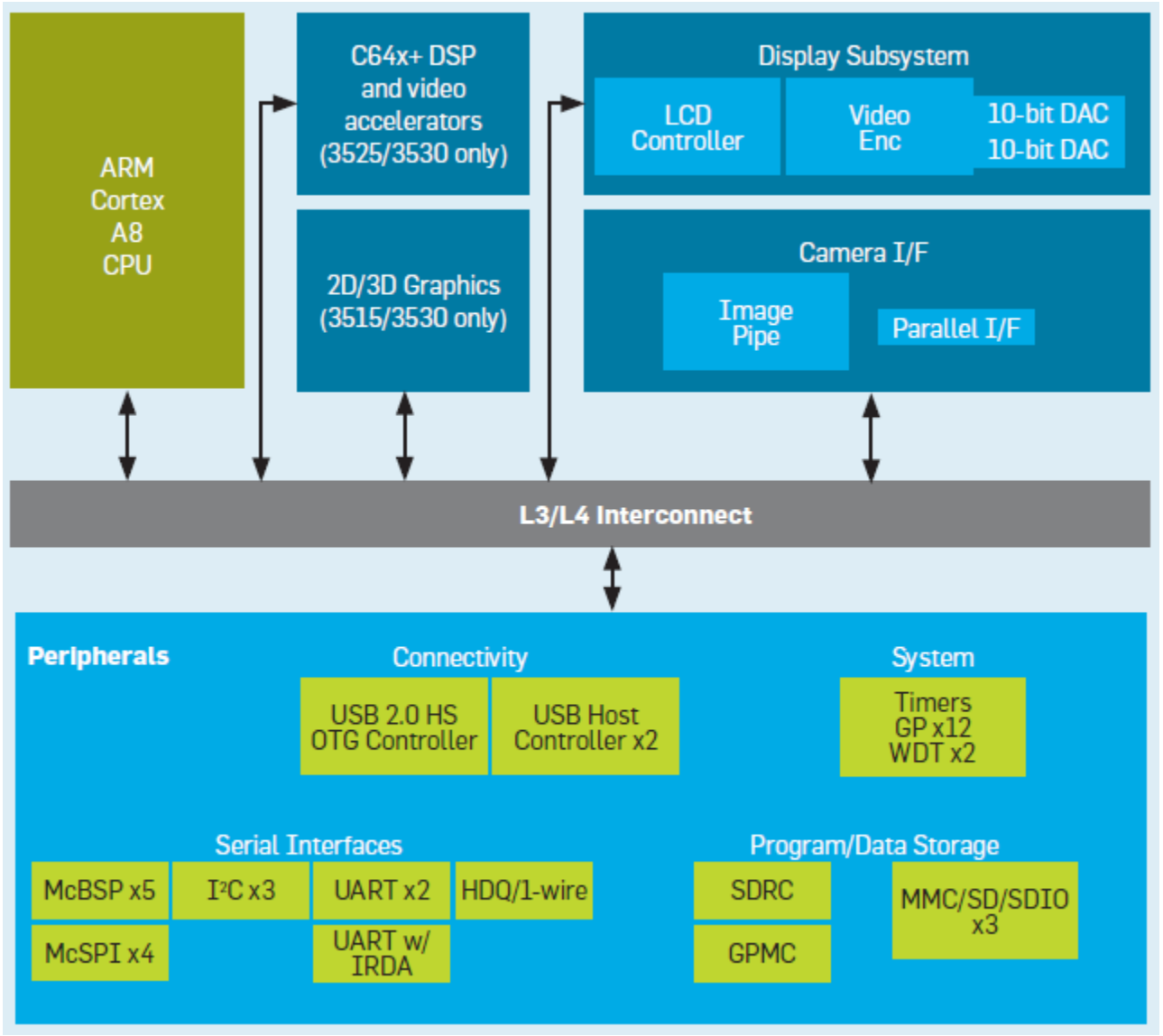
- Package Power/total Energy consumption limits number of Transistors
  - Adding more cores & operating chips at highest frequency power consumption would be prohibitive
  - Keeping power within reasonable bounds severely limits improvement in microprocessor performance



Optimization

- Traditional approach: invest max transistors in the 90% case to increase single-thread performance that can be applied broadly
- New scaling regime (slow transistors, energy improvements)
  - Makes no sense to add transistors to a single core as energy efficiency suffers
  - Using additional transistors to build more cores produces limited benefit
    - Increased performance apps w/ thread parallelism
- 90/10 optimization no longer applies – Instead
  - Optimizing w/ a 10% accelerator for a 10% case, then another for different 10% case, then another … often produces a system w/ better overall energy efficiency and performance -> “10 x 10 optimization”
  - Operating the chip w/ 10% of transistors active / 90% inactive, but a different 10% at each point in time
  - 20 generations into Moore’s Law have shifted the balance
    - Using a fraction of transistors on chip seems to be the right solution

System-on-a Chip (TI)



Energy Efficiency is Driver for Performance

- Reality of finite energy budget for processors must produce quantitative shift in how chip architects think
  - Energy efficiency is key metric for design
  - Energy-proportional computing must be ultimate goal for H/W architecture and software-application design
- While this is recognized in macro-scale computing (large-scale data centers), the idea of micro-scale energy-efficient computing in Microprocessors is even more challenging
  - With fixed energy budget this corresponds directly to higher performance
  - Quest for extreme energy efficiency is ultimate driver for performance

Software Challenges renewed: Programmability vs. Efficiency

- End of scaling of single-thread performance means major software challenges
  - Shift to symmetric parallelism caused greatest software challenge in history of computing
  - Pressure on energy-efficiency will lead to extensive use of heterogeneous cores and accelerators
  - Need/need to adopt high-level “productivity” languages built on advanced interpretive and compiler technologies and increasing use of dynamic translation techniques
- Trend: higher-level programming, extensive customization through libraries, sophisticated automated performance search techniques (i.e. autotuning)

Logic Organization Challenges, Trends, Directions

Challenge	Near-Term	Long-Term
Integration and memory model	I/O-based interaction, shared memory spaces, explicit coherence management	Intelligent, automatic data movement among heterogeneous cores, managed by software-hardware partnership
Software transparency	Explicit partition and mapping, virtualization, application management	Hardware-based state adaptation and software-hardware partnership for management
Lower-power cores	Heterogeneous cores, vector extensions, and GPU-like techniques to reduce instruction- and data-movement cost	Deeper, explicit storage hierarchy within the core; integrated computation in registers
Energy management	Hardware dynamic voltage scaling and intelligent adaptive management, software core selection and scheduling	Predictive core scheduling and selection to optimize energy efficiency and minimize data movement
Accelerator variety	Increasing variety, library-based encapsulation (such as DX and OpenGL) for specific domains	Converged accelerators in a few application categories and increasing open programmability for the accelerators

Things that may Change

Due to energy consumption constraints

- Drop H/W support for single flat address space, single-memory hierarchy, steady rate of execution
- Future systems will place components under S/W control
  - Requires sophisticated S/W tools to manage H/W boundaries and irregularities w/ greater energy efficiency
  - High-performance applications may manage these complexities explicitly
  - Architectural features shift responsibility for distribution of computation and data across compute and storage elements of Microprocessors to software
  - Improves energy efficiency but requires advances in applications, compilers, runtime environments and OSs to understand and predict application/workload behavior
    - Advances require radical research breakthroughs & major changes in S/W practices (next slide)

Software Challenges, Trends, Directions

Challenge	Near-Term	Long-Term
1,000-fold software parallelism	Data parallel languages and "mapping" of operators, library and tool-based approaches	New high-level languages, compositional and deterministic frameworks
Energy-efficient data movement and locality	Manual control, profiling, maturing to automated techniques (auto-tuning, optimization)	New algorithms, languages, program analysis, runtime, and hardware techniques
Energy management	Automatic fine-grain hardware management	Self-aware runtime and application-level techniques that exploit architecture features for visibility and control
Resilience	Algorithmic, application-software approaches, adaptive checking and recovery	New hardware-software partnerships that minimize checking and recomputation energy

Conclusions - Microprocessors

- Great old days of Moore's Law scaling delivered 1,000 fold performance improvement in 20 years
- Pretty good new days will be more difficult
  - Frequency of operation will increase slowly
  - Few large cores, large number of small cores operating at low frequency and low voltage
  - Aggressive use of accelerators will yield highest performance
  - Efficient data orchestration w/ more efficient memory hierarchies and new types of interconnects tailored for locality will be critical
    - Depends on sophisticated S/W to place computation and data so as to minimize data movement
  - Programming Systems will have to comprehend restrictions and provide Tools & Environments to harvest the performance
- Use of materials & technologies other than Si CMOS for processor design will face challenges rendering the ones we're fighting with today a warm-up exercise for what lies ahead

Data Storage and Data Access

- POSIX I/O becoming serious Impediment to I/O performance and scaling
  - Hasn't changed much in 26 years though world of storage evolves
  - Change or relaxation could spur development of new storage mechanisms to improve application performance, management, reliability, portability, and scalability
- ODMS not successful but provided valuable lessons
- Object Storage Technology
  - Scaling shared storage to extraordinarily levels of b/w & capacity w/o sacrificing reliability or simple, low cost operations
  - Key properties include variable length ordered sequence of addressable bytes, embedded management of data layout, extensible attributes and fine grain device-enforced access control

Resource Provisioning

- HEP facing Transition to more shared and Opportunistic Resources provide through variety of Interfaces
  - Effective use of diverse environments
  - Perform provisioning of variety of different processing and storage resources across dedicated, specialized, contributed, opportunistic and purchased resources
- Compared to present situation Computing in HEP will become a much less Static System
  - Resource Providers will be continuously flowing in and out of Distributed Computing Environments
- Key Issue is the Time it needs to register Resources and adopt Applications
  - Should be no longer than 10% of Execution Time
  - APIs are key for creating new composite applications w/o developer needing to know anything about underlying compute, network, and storage resources. Application APIs make SaaS possible

Networking

Next gen advanced Networked Apps will require network capabilities and services far beyond what's available from networks today

- Dynamic network service topologies (overlays) with replication capabilities
- Resource management and optimization algorithms (e.g. available network resources, load on depots)
- Use of multi-constraint path finding algorithms to optimize solutions
- Network caching to optimize transfers and data access
  - Let the network learn what jobs need (assuming overlapping interest)
  - Dynamic, self-learning caches at major network exchange points
  - "Named Data Networks" support content distribution
    - Requests are routed based on content rather than file names mapped to IP addresses